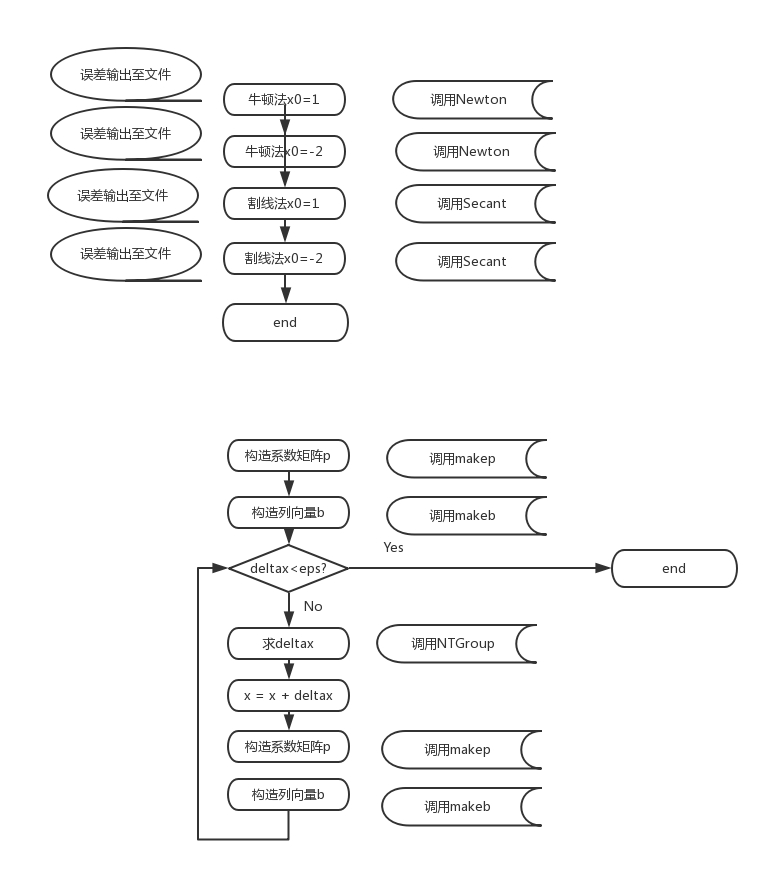
1. 分别使用牛顿法和割线法求解方程的根,2.使用牛顿法求非线性解方程组

程序流程图如下



注：

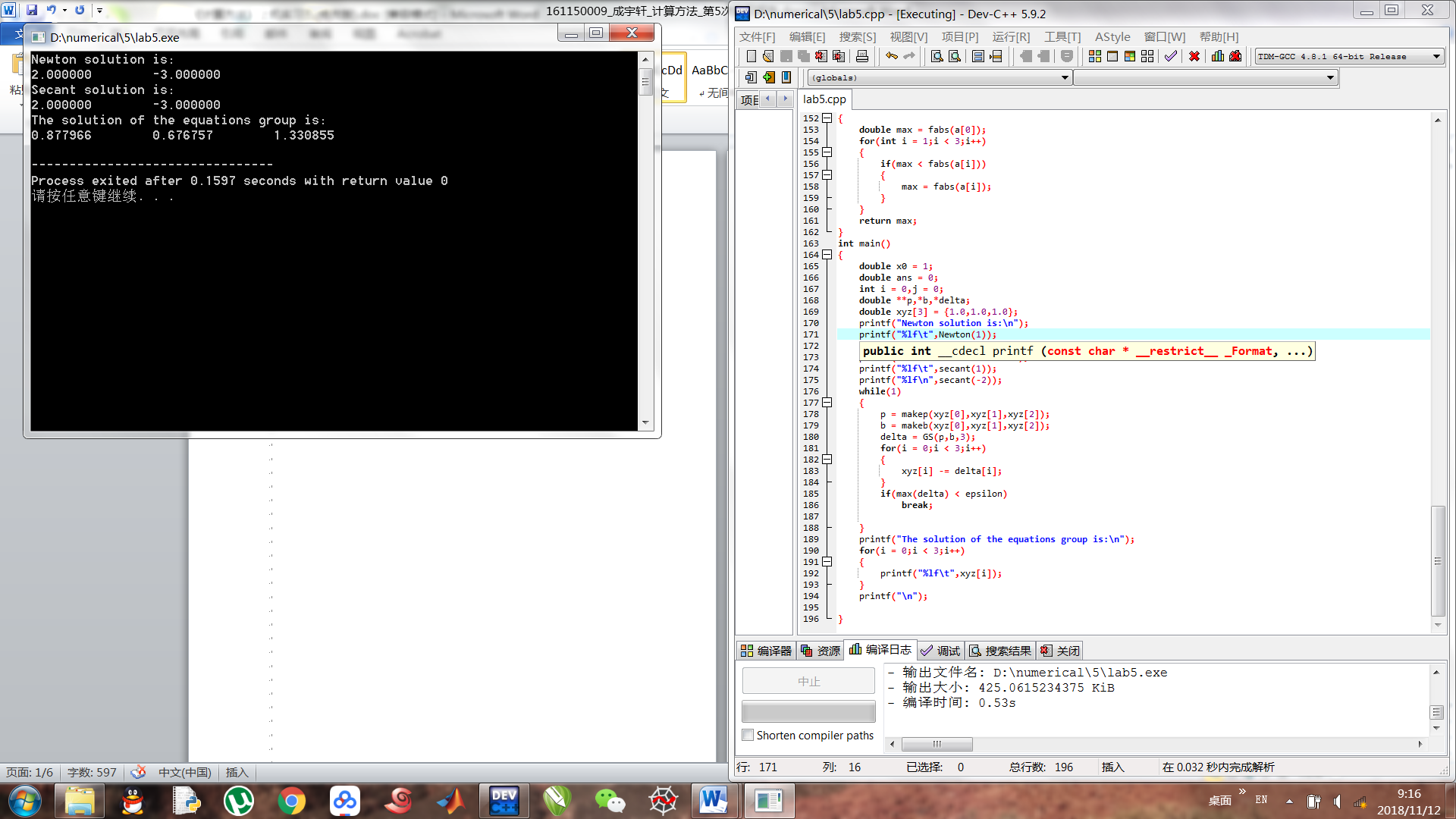
NTGroup 应为 GS ，使用高斯消元法解矩阵方程

运行结果截图如下：

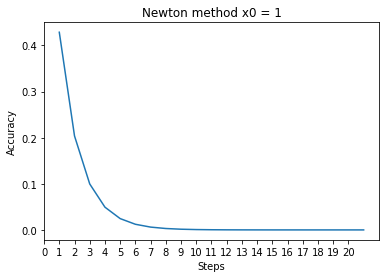
首先输出牛顿法的解，初始x分别为1，-2，解为 2，-3

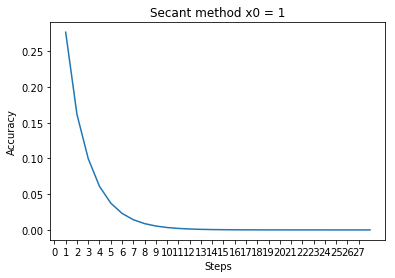
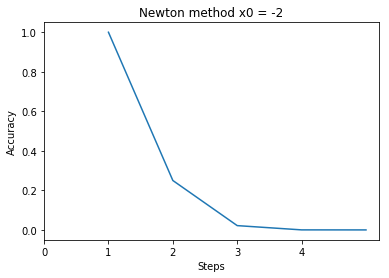
随后输出割线法的解，初始x分别为1，-2，解为 2，-3

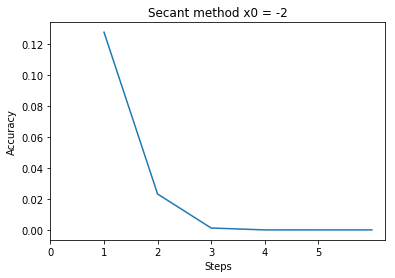
最后输出方程组牛顿法的解，初始向量为（1，1，1），解为0.877966，0.676757，1.330855



将误差随迭代次数的曲线画出，此处使用python实现







可见，迭代的误差随迭代次数的增加呈近似指数型下降的趋势，说明迭代法是收敛的。分别对两种方法进行比较，割线法的误差较牛顿法的小.

对于重根来看，原方程有一重根x=2，两方法的对此 的迭代次数均为20多次，而单根x=-3所需迭代次数仅为4、5次，二者均从差1的初始值进行迭代。从方法本身出发，由于在重根附近的斜率较小，逐渐为0，而在单根处并不存在此条件，即二阶导为0. 所以收敛速度在重根处会较单根处慢。

绘图所用python代码如下：

# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 12 09:24:00 2018

@author: Administrator

"""

from matplotlib.ticker import MultipleLocator

import matplotlib.pyplot as plt

import xlrd

if \_\_name\_\_ == "\_\_main\_\_":

workbook = xlrd.open\_workbook(r'C:\lab5.xlsx')

sheet1 = workbook.sheet\_by\_index(0)

data = [[]]

for i in range(0,4):

x = sheet1.col\_values(2\*i)

y = sheet1.col\_values(2\*i + 1)

while(x[-1] == ''):

x.pop(-1)

while(y[-1] == ''):

y.pop(-1)

data.append([])

data[i].append(x)

data[i].append(y)

fig = []

for i in range(0,4):

p = plt.figure()

plt.xlabel("Steps")

plt.ylabel("Accuracy")

plt.plot(data[i][0],data[i][1])

plt.xticks(range(0,len(data[i][0])))

if i < 2:

title = "Newton method "

else:

title = "Secant method "

if i % 2 == 0:

title += "x0 = 1"

else:

title += "x0 = -2"

plt.title(title)

fig.append(p)

程序C语言源代码如下：

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#define epsilon 1e-6

double \*GS(double \*\*a,double \*b,int n)//gaosi xiaoyuan

{

double \*\*ans = (double\*\*)malloc(n\*sizeof(double\*));

double \*\*para = (double\*\*)malloc(n\*sizeof(double\*));

double \*bb = (double\*)malloc(n\*sizeof(double));

double suml,sumu;

int i,j,k;

for(i = 0;i < n;i++)

{

ans[i] = (double\*)malloc(n\*sizeof(double));

para[i] = (double\*)malloc(n\*sizeof(double));

}

for(i = 0;i < n;i++)

{

for(j = 0;j < n;j++)

{

ans[i][j] = a[i][j];

}

bb[i] = b[i];

}

for(i = 0;i < n;i++)

{

for(j = 0;j < n;j++)

{

if(i != j)

{

for(k = 0;k < n;k++)

{

if(k != i)

{

ans[j][k] = -ans[j][i] / ans[i][i] \* ans[i][k] + ans[j][k];

}

}

bb[j] = bb[j] + bb[i] \* -ans[j][i] / ans[i][i];

ans[j][i] = 0;

}

}

}

for(i = 0;i < n;i++)

{

bb[i] = bb[i] / ans[i][i];

}

return bb;

}

double f(double x){return pow(x,3) - pow(x,2) - 8\*x + 12;}

double ff(double x){return 3\*pow(x,2) - 2\*x - 8;}

double Newton(double xs)

{

double xe = xs - f(xs)/ff(xs);

double temp = xs,ans = 0;

char cBuffer[50];

int count = 1;

FILE \*fp;

sprintf(cBuffer,"C:\\newtonxstart\_%lf.txt",xs);

fp = fopen(cBuffer,"w+");

while(fabs(xe - xs) > 1e-7)

{

xs = xe;

xe = xs - f(xs)/ff(xs);

}

ans = xe;

xs = temp;

xe = xs - f(xs)/ff(xs);

while(fabs(xe - xs) > 1e-7)

{

fprintf(fp,"%d\t%lf\n",count,fabs(xe - ans));

xs = xe;

xe = xs - f(xs)/ff(xs);

count++;

}

fprintf(fp,"%d\t%lf\n",count,fabs(xe - ans));

fclose(fp);

return ans;

}

double secant(double xs)

{

double xe = xs - f(xs)/ff(xs);

double k,y,x;

double temp = xs,ans = 0;

char cBuffer[50];

int count = 1;

FILE \*fp;

sprintf(cBuffer,"C:\\secantxstart\_%lf.txt",xs);

k = (f(xs) - f(xe)) / (xs - xe);

x = -f(xs) / k + xs;

while(fabs(xe - x) > 1e-7)

{

k = (f(x) - f(xe)) / (x - xe);

xs = x;

x = -f(xe) / k + xe;

xe = xs;

}

ans = xe;

fp = fopen(cBuffer,"w+");

xs = temp;

xe = xs - f(xs)/ff(xs);

k = (f(xs) - f(xe)) / (xs - xe);

x = -f(xs) / k + xs;

while(fabs(xe - x) > 1e-7)

{

k = (f(x) - f(xe)) / (x - xe);

xs = x;

x = -f(xe) / k + xe;

xe = xs;

fprintf(fp,"%d\t%lf\n",count,fabs(x - ans));

count++;

}

return ans;

}

double f1(double x,double y ,double z){return 16\*pow(x,4) + 16\*pow(y,4) + pow(z,4) - 16;}

double f2(double x,double y ,double z){return pow(x,2) + pow(y,2) + pow(z,2) - 3;}

double f3(double x,double y){return pow(x,3) - y;}

double ff1xy(double x){return 64\*pow(x,3);}

double ff1z(double z){return 4\*pow(z,3);}

double ff2xyz(double x){return 2\*x;}

double ff3x(double x){return 3\*pow(x,2);}

double \*makeb(double x,double y ,double z)

{

double \*b;

b = (double\*)malloc(3\*sizeof(double));

b[0] = f1(x,y,z);

b[2] = f2(x,y,z);

b[1] = f3(x,y);

return b;

}

double \*\*makep(double x,double y ,double z)

{

double \*\*p = (double\*\*)malloc(3\*sizeof(double\*));

for(int i = 0;i < 3;i++)

{

p[i] = (double\*)malloc(3\*sizeof(double));

}

p[0][0] = ff1xy(x);

p[0][1] = ff1xy(y);

p[0][2] = ff1z(z);

p[2][0] = ff2xyz(x);

p[2][1] = ff2xyz(y);

p[2][2] = ff2xyz(z);

p[1][0] = ff3x(x);

p[1][1] = -1;

p[1][2] = 0;

return p;

}

double max(double \*a)

{

double max = fabs(a[0]);

for(int i = 1;i < 3;i++)

{

if(max < fabs(a[i]))

{

max = fabs(a[i]);

}

}

return max;

}

int main()

{

double x0 = 1;

double ans = 0;

int i = 0,j = 0;

double \*\*p,\*b,\*delta;

double xyz[3] = {1.0,1.0,1.0};

printf("Newton solution is:\n");

printf("%lf\t",Newton(1));

printf("%lf\n",Newton(-2));

printf("Secant solution is:\n");

printf("%lf\t",secant(1));

printf("%lf\n",secant(-2));

while(1)

{

p = makep(xyz[0],xyz[1],xyz[2]);

b = makeb(xyz[0],xyz[1],xyz[2]);

delta = GS(p,b,3);

for(i = 0;i < 3;i++)

{

xyz[i] -= delta[i];

}

if(max(delta) < epsilon)

break;

}

printf("The solution of the equations group is:\n");

for(i = 0;i < 3;i++)

{

printf("%lf\t",xyz[i]);

}

printf("\n");

}